**POINTERS**

**Example**

char \*ptr = Null;

s1 = “Amit,Varun,asha”;

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | m | i | t | , | V | A | R | . | . |
| 2000 | 2001 | 2002 | 2003 | 2004 | 2005 | . | . | . | . |

ptr = s1; ----->>>>>>>>>>> it will point to address of s1, ptr = 2000

printf(“%c”,\*ptr) ----->>>>>>>>>>> A (it points to base address of s1, value(2000) -> and its value is A character)

ptr++; (now pointing address will be incremented by 1) ptr = 2001

putchar(\*ptr) ----->>>>>>>>>>> m ---> value(2001).

1. //Accessing string using delimiter

#include <stdio.h>

int main() {

char lines[] = "Amit,Kumar,Ravi";

char Names[10][20];

char \*ptr = NULL;

ptr = lines;

int row =0;

int col =0;

while(\*ptr!='\0')

{

putchar(\*ptr);

ptr++;

}

ptr = lines;

while(\*ptr!='\0')

{

if(\*ptr==',')

{

Names[row][col] == '\0';

break;

}

Names[row][col] = \*ptr;

ptr++;

col++;

}

putchar(\*ptr);

puts(Names[row]);

ptr++;

putchar(\*ptr);

return 0;

}

2.

#include <stdio.h>

int main() {

char lines[] = "Amit te,Kumar,Ravi122ww,dhhd,hds,dhg,hs,sh,hb,bsx,sgdsdh";

char Names[10][20];

char \*ptr = NULL;

ptr = lines;

int row =0;

int col =0;

while(\*ptr!='\0')

{

putchar(\*ptr);

ptr++;

}

ptr = lines;

while(\*ptr!='\0')

{

if(\*ptr==',')

{

Names[row][col] == '\0';

break;

}

Names[row][col] = \*ptr;

ptr++;

col++;

}

putchar(\*ptr);

puts(Names[row]);

while(1)

{

ptr++;

row++;

col=0;

if(\*ptr == '\0')

{

break;

}

while(\*ptr!='\0')

{

if(\*ptr==',')

{

Names[row][col] == '\0';

break;

}

Names[row][col] = \*ptr;

ptr++;

col++;

}

putchar(\*ptr);

puts(Names[row]);

}

ptr++;

putchar(\*ptr);

return 0;

}

Structures

We want to create multiple of its type – so we need user defined datatype --- Structure.

struct tagName{

mem/properties of structure

};

* Structure within a structure can also be defined.

Example:

typdef struct Employee

{

int id;

int sal;

int phno;

char Name[20];

char Gender;

}EMP;

**TO ACCESS:**

In main code : EMP e1;

e1.id

**TO ASSIGN VALUE:**

strcpy(e1.Name,”Bhima”) // for strings

e1.Gender = ‘M’;

***Assigning value from some other array.***

e1.id = atoi(Names[0]); // Here Names[0] is string, but we need integer. So use atoi to change string to int datatype

strcpy(e1.Name,Names[1]);

e1.Gender = Names[2][0]; here we need to get only character , so use col value.

e1.phno = atoi(Names[3]);

e1.sal = atoi(Names[4]);

***To display the employeeDetails.***

Int display(EMP \*e)

{

If(e==NULL)

Return 1;

Prinf(“\nID: %d”,e -> id); // in function argument it is EMP e, then it should be e.id in printf statement

// and all details similarly

Return 0;

}

// prototype this function after struct definition.

// while calling this function ----->>>> it should be display(&e); // here ‘&’ is used, because it is static array. If it is dynamic array, we should not use ‘&’

EMP \*testEmp = NULL;

testEmp = (Emp \*)malloc(sizeof(EMP)) ; // it will allocates space to testEmp.

**DYNAMIC MEMORY ALLOCATION**

: It points to unnamed address

**Example:**

EMP e1; // e1 is named address

EMP \*e; // e is pointing to unmaned address

1. e = &e1; // Now e1 points to named address i.e.e1

***To scan ‘e1’ data***

scanf(“%d%d%d”, &e1.id,&e1.sal,&e1.phno);

scanf(“%s”,e1.name);

getchar(); to free

scanf(“%c”,&e1.Gender);

***To print ‘e’ data***

Printf(“\nID: %d”,e -> id);

Similarly all

1. e = (EMP \*)malloc(sizeof(EMP)); // unnamed address

***To scan ‘e’ data***

scanf(“%d%d%d”, &e->id,&e->sal,&e->phno);

scanf(“%s”,e->name);

getchar(); to free

scanf(“%c”,&e->Gender);

***To print ‘e’ data***

Printf(“\nID: %d”,e -> id);

Similarly all

free(e); // we have to use free( ) if we allocated dynamically an array to deallocate.

1. **To have n number of employees**

Int noEmp;

Scanf(“%d”,&noEmp);1

e = (EMP \*)malloc(noEmp \* sizeof(EMP));,

temp = e;

**To getEmpDetails**

For(i=0;i<noEmp;i++;e++)

getDetails(e);

e = temp;

**To DisplayDetails**

For(i=0;i<noEmp;i++;e++)

DispDetails(e);

Free(e); // here we will get error as invalid pointer, because it is pointing to null character which is not reserved.